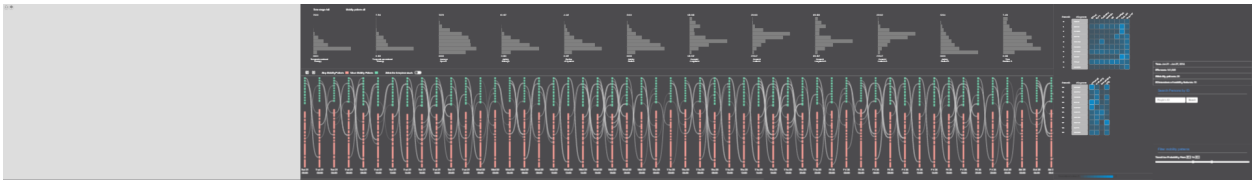
Weekly Report

11/07/2016-11/13/2016

Work

- I have reviewed a paper: Delivering Real-time Information Services on Public Transit: a Framework.
- I extract 84 graphs with 4407 nodes from mobile trajectory data. Each node represents a basestation. If there is a movement from station A to station B, I put two nodes into the graph and add an edge between two nodes. I calculate graphs per two hours for one week to get 7*24=84 graphs. I still have some problems such as running out of memory, because each graph needs 4407*4407=19421649 Int type to store. 84 graphs consume about 28GB memory. I will store graphs into a sparse matrix library later. Also, I output graphs into a CSV file (3.5GB) and embed them into 2D dimension using LargeVis on master node of our cluster (about 1 hour). Next, I will draw a scatterplot to explore the result.
- This week, we try to transform our AMTG website form PC to Large Screen.



Plan for next week

- Extract graphs from taxi GPS trajectory.

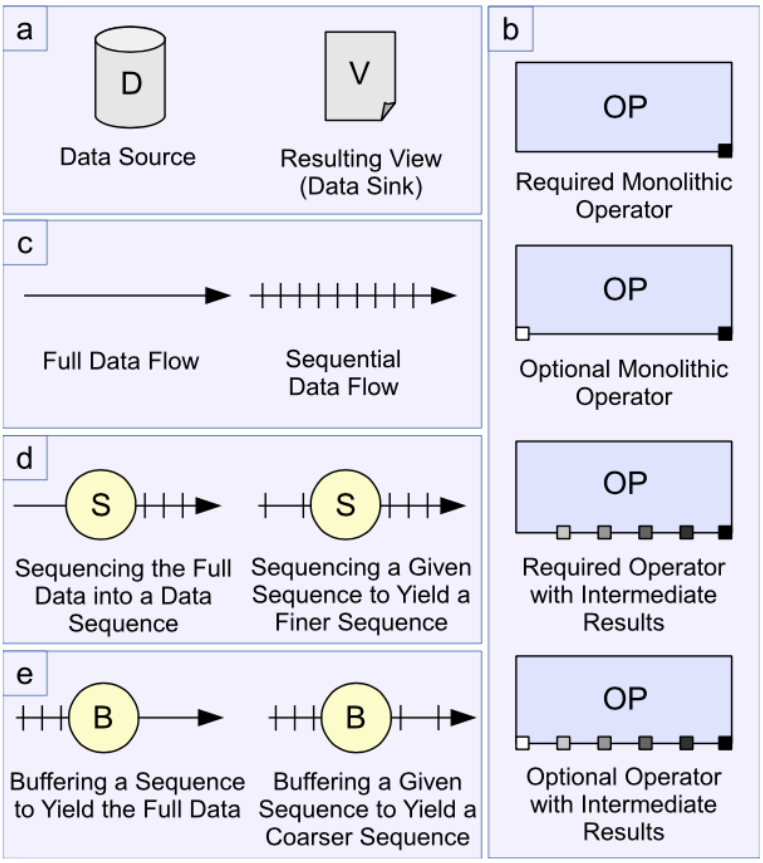
- Read some deep learning papers about graph.

1. Delivering Real-time Information Services on Public Transit: a Framework

本文主要是设计了一个分布式数据处理系统，可以将城市数据，如地图，轨迹，交通路线，社交数据，无论是静态的还是实时流获取的都可以预处理后存储到数据仓库中。同时，系统根据数据仓库的数据，提供一些分析数据接口，比如公共交通实时位置，社交网络关于交通拥堵等信息的消息，以及公众可以从中统一的获取城市数据进行分析。每当有新数据流进来时，系统会首先更新数据仓库，再同时数据分析模块更新显示的结果。所以，这个系统覆盖了城市数据分析从获取存储到分析的每一个步骤。

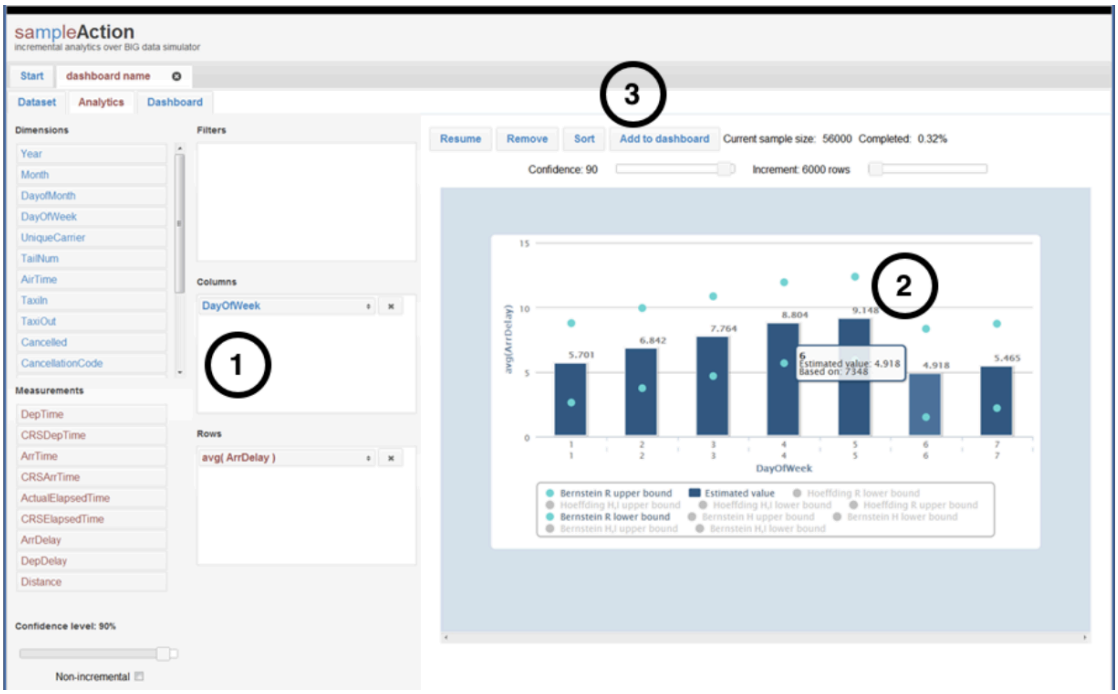
2. An Enhanced Visualization Process Model for Incremental Visualization

当前由于数据量过多过于复杂，传统的方法无法有效的可视化数据。而增量式可视化却可以解决这个问题，他允许用户在可视化的中间过程中就对可视化的结果进行改变。作者提出了 Date state Reference Model，用来展示数据和可视化操作。



3. Trust Me, I'm Partially Right: Incremental Visualization. Lets Analysts Explore Large Datasets Faster.

类似于 2，本文也是处理超大规模数据集的一种方法。由于对大规模数据的查询过慢，所以对于人可视化探索的过程有着重大的阻碍。所以文章采用对整个数据集采样的方法，总而支持实时的可视化操作。（3）中显示了当前采样的进度。


sampleAction
incremental analytics over BIG data simulator
Start
dashboard name
Dataset
Analytics
Dashboard
Dimensions
Year
Month
DayofMonth
DayOfWeek
UniqueCarrier
TailNum
AirTime
TaxiIn
TaxiOut
Cancelled
CancellationCode
Measurements
DepTime
CRSDepTime
ArrTime
CRSArrTime
ActualElapsedTime
CRSElapsedTime
ArrDelay
DepDelay
Distance
Confidence level: 90%
Non-incremental
Filters
Columns
DayOfWeek
1
Rows
avg(ArrDelay)
3
Resume
Remove
Sort
Add to dashboard
Current sample size: 56000 Completed: 0.32%
Confidence: 90
Increment: 6000 rows
2
avg(ArrDelay)
15
10
5
0
5.701
6.842
7.764
8.804
9.148
4.918
5.465
6
Estimated value: 4.918
Based on: 7348
1 1
2 2
3 3
4 4
5 5
6 6
7 7
DayOfWeek
Bernstein R upper bound
Estimated value
Hoeffding R lower bound
Hoeffding H,I upper bound
Hoeffding H,I lower bound
Hoeffding R upper bound
Bernstein R lower bound
Bernstein H upper bound
Bernstein H lower bound
Bernstein H,I upper bound
Bernstein H,I lower bound

4. Semantic trajectories: Mobility data computation and annotation.

这篇文章提出了 SemanticTraj 的可视分析系统。通过把轨迹数据转化为文本描述数据，通过文本搜索的交互方式对轨迹数据进行查询分析。一方面，通过数据文本搜索的查询方式不需要分析人员进行专业训练，因为我们比较熟悉百度、Google 这样的搜索引擎。另一方面，把轨迹数据转化为文本描述能提供高层次的信息，更容易帮助用户总结分析结果。

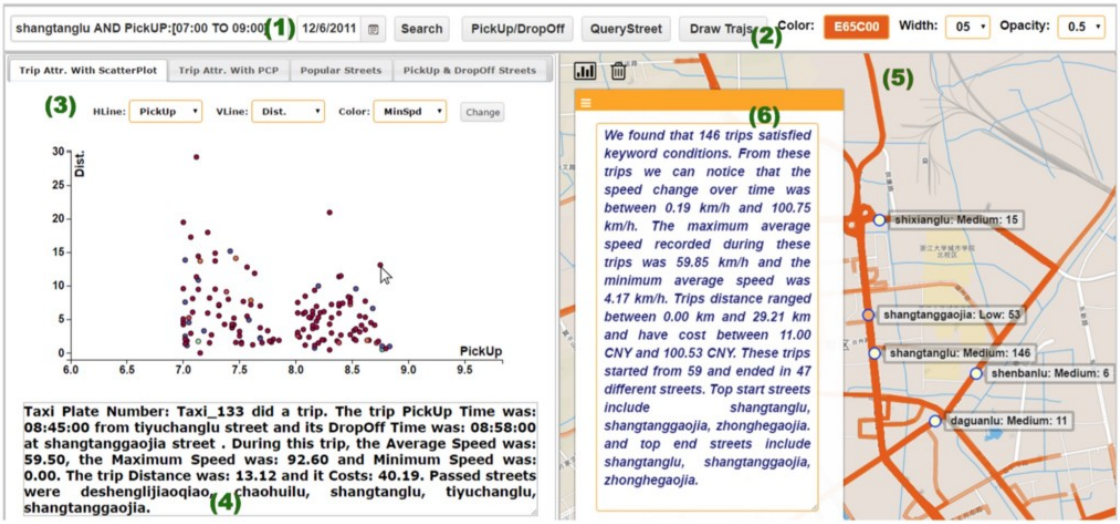


Fig. 3. *Using SemanticTraj to visualize taxi trips which passed Shangtanglu street of Hangzhou, China in the morning (7am-9am) of Dec 6, 2011. See details in Sec. 7. (1) Query input box accepting semantic query conditions as Shangtanglu AND PickUp:[7:00-9:00]; (2) Visualization control panel for adjusting the appearance; (3) Scatterplot view for users to study search results. Other visual tools can be selected in this view; (4) Meta-summary of a selected trip which automatically summarizes the trip fact; (5) Map view showing trip trajectories. Text labels are displayed on critical streets about its role in these trips; (6) A meta-summary of the group of all 146 result trips. Users can interact with the name tags to filter trips.*

5. TrajGraph: A Graph-Based Visual Analytics Approach to Studying Urban Network Centralities Using Taxi Trajectory Data

这周又重新阅读了一下这篇文章，参考了这样的算法进行构造图，但由于基站的节点过多就造成了计算的缓慢。

1. for each trajectory T_j do
2. for each GPS sample point Pt of T_j in T do
   {
   read next sample point on T_j
3. nextPt = Pt.nextPoint(); if nextPt is null, go to 1;
4. cID = Pt.roadID; nID = nextPt.roadID;
5. if (cID == nID), go to 2;
   remove incorrect consecutive samples
6. if (Distance(Pt, nextPt)> ξ_d), go to 2;
7. if (TimeDifference(Pt, nextPt)> ξ_t), go to 2;
   update graph
8. add a new vertex N_{cID} to G_T when it is not in G_T;
9. add a new vertex N_{nID} to G_T when it is not in G_T;
10. add a directed edge $E_{cID,nID}$ to G_T when it is not in G_T;
11. update the weight of N_{cID} with proper methods (Sec. 4.1.2);
    }

这篇文章为了达到实时响应的目的，对节点进行了合并，利用图分割算法。